Rapport : Pré-projet

MINIX 3

Le 19 avril 2009

Lamouline Laurent
3597-05-00

1. Le fichier "get_noma.h"

Tout d'abord, il était nécessaire de créer une librairie contenant l'en-tête de l'appel système à réaliser. Cette dernière contient dès lors la ligne suivante:

```
_PROTOTYPE( int get_noma, (struct noma buffer));
```

On définit donc ici le nom de l'appel (*get_noma*), le type renvoyé (*int*) et le type d'argument (*struct noma buffer*). La structure est également définie dans ce fichier et à la forme suivante:

```
typedef struct{
    int noma_nr;
} noma;
```

Cette structure contient un entier (*noma_nr*) représentant le numéro de noma car j'ai choisi d'implémenter un appel système écrivant mon numéro de noma dans la structure. Cette librairie est alors placée dans le répertoire */include/sys* de MINIX afin que l'en-tête de l'appel soit connu.

2. Le fichier "_get_noma.c"

Après avoir écrit la librairie permettant au système de reconnaître mon appel système, il est maintenant nécessaire d'écrire une fonction _get_noma.c qui fera le lien entre la couche utilisateur et la couche des serveurs lorsque on fera l'appel système. La communication sous MINIX étant essentiellement basée sur les messages, nous allons donc utiliser la technique du message passing afin de faire communiquer les processus. Cette technique va me permettre de réaliser l'appel système. En effet, on envoie un message (de type *message m;*) au PM via l'appel *_syscall(PM, GET_NOMA, &m)*, ce message contient l'adresse virtuelle de l'endroit mémoire où l'on souhaite écrire le noma ainsi que l'identification du processus à l'origine du message (et le timestamp). Le timestamp et l'id du processus expéditeur sont automatiquement placé dans le message par le kernel.

3. Le fichier "get_noma.s"

Ce fichier contient des instructions en langage d'assemblage. Lorsque l'appel à *get_noma()* est fait, c'est en fait *_get_noma()* qui est appelé, c'est à ca que sert le fichier *get_noma.s*, il fait le lien.

4. Le fichier "get_noma.c"

C'est ici qu'est reçu le message envoyé dans le fichier *_get_noma.c*. On récupère alors le numéro du processus expéditeur du message grâce au champ *m_in.m_source* du message et l'adresse virtuelle où l'on souhaite écrire en mémoire via le champ *m_in.m1_p1*. Grâce à cela, on va pouvoir faire appel à l'appel système *sys_datacopy* pour copier *nbrtocopy*, représentant la donnée à copier (mon noma: 35970500), à l'adresse mémoire récupérée précédemment. C'est alors le kernel qui va s'occuper de traduire l'adresse virtuelle en adresse physique et qui va effectuer l'écriture.

5. Le fichier "test.c"

C'est bien entendu dans ce fichier (placé dans le répertoire test) que je vais tester l'appel système qui a été implémenté comme décris ci-dessus. Il consiste en la déclaration d'un pointeur vers une structure de type noma et de l'appel suivant: `get_noma(test);`
Ensuite on vérifie que l'appel a correctement fonctionné en faisant: *printf("Valeur contenue dans test->noma_nr: %d", test->noma_nr);*
On constate que c'est bien la valeur 35970500 (*mon noma* ☺) qui est affichée donc ca fonctionne.

6. Fichiers MINIX 3 modifiés

Afin d'intégrer correctement l'appel système au système MINIX 3, il a été nécessaire de modifier certains fichiers qui étaient déjà présents. L'un d'entre eux est *table.c* présent à la fois dans *servers/pm* et *servers/fs*. En effet, ce fichier contient une table de correspondance entre les appels système et le numéro des routines qui les réalisent (*do_get_noma* a le numéro 95), il a donc fallu ajouter la correspondance routine – n° pour que le système puisse l'identifier.
Il a également été nécessaire de modifier le fichier *proto.h* présent dans *servers/pm* afin de définir le prototype de *do_get_noma*.
Il a bien entendu également été nécessaire de mettre les différents *Makefile* à jour afin que les fichiers qui ont été ajoutés soient pris en compte lors de la compilation via les commandes:

- make libraries (pour compiler les *includes*)
- make hdboot (pour compiler entre autres *servers/pm* et *servers/fs*)